Manuel d'Introduction au Logiciel RATS

Laurent Ferrara[1]

Octobre 2004
Version 2

[1] Centre d'Observation Economique, 27 avenue de Friedland, 75382 Paris Cedex 08.
mél : lferrara@ccip.fr
site : http//lo.ferrara.free.fr

Plan

Introduction

Le but de ce manuel est d'initier le praticien, statisticien ou économètre, à l'utilisation du logiciel RATS (*Regression Analysis of Times Series*). Nous considérons ici la version 5 du logiciel, tout en sachant qu'une version 6 vient d'être diffusée récemment. Toutefois, les procédures présentées dans ce document fonctionnent avec la nouvelle version ainsi qu'avec l'ancienne version 4 du logiciel. Dans ce manuel, nous ne rentrons pas dans le détail des méthodes statistiques et économétriques de traitement des séries temporelles, et nous renvoyons le lecteur déjà connaisseur des fonctionnalités du logiciel au polycopié intitulé "Econométrie des séries chronologiques avec le logiciel RATS" disponible sur le site Internet http://lo.ferrara.free.fr dans la rubrique Rats.

Le logiciel RATS est un logiciel entièrement dédié à l'étude des séries chronologiques et se distingue ainsi des autres logiciels de statistiques présents sur le marché, tels que SAS, S-Plus ou SPSS. L'objet de base manipulé dans ce logiciel est donc une série chronologique. Cependant, RATS permet également d'effectuer, entre autres, du calcul matriciel, ce qui élargit considérablement sa palette des méthodes statistiques utilisables.

Le logiciel RATS possède un nombre important de fonctions et d'instructions internes, qui en font un des logiciels de traitement des séries chronologiques les plus performants. Cependant, un intérêt fondamental de RATS est qu'il s'agit d'un logiciel complètement ouvert. A savoir, l'utilisateur est libre de créer ses propres procédures. De nombreux chercheurs universitaires utilisent ce logiciel, et un certain nombre d'entre eux met à la disposition du public ses procédures, ce qui permet d'enrichir constamment la bibliothèque des méthodes statistiques de traitement des séries chronologiques, utilisables avec RATS. Nous donnons à la fin de cette introduction une liste de quelques sites Internet mettant en ligne des procédures RATS.

Ce manuel ne prétend pas être exhaustif et ne détaille pas toutes les instructions RATS, mais propose plutôt d'initier le néophyte à la manipulation du logiciel. On se réfère systématiquement au manuel d'utilisation de RATS (Doan, 1992), fourni par la société Estima qui distribue le logiciel, pour une description fouillée des instructions. Lors d'une session RATS, on recommande vivement l'utilisation du manuel d'utilisation de RATS, ainsi que l'utilisation systématique de l'aide en ligne du logiciel, qui est assez bien faite.

Les références bibliographiques concernant le traitement des séries chronologiques avec RATS ne sont pas nombreuses, mais nous en donnons quelques-unes ci-dessous :

Doan, T.A. (1992), *RATS User's Manual*, Estima, Evanston, IL.

Enders, E. (1996), *RATS Handbook for Econometric Time Series,* Wiley, New York.

Enders, E. (2003), *RATS Programming Manual,*distribué gratuitement par Estima sur leur site web à l'adresse : www.estima.com

McCullough, B.D. (1997), "A review of RATS v4.2: Benchmarking numerical accuracy", *Journal of Applied Econometrics*, 12, 181-190.

Nous avons souligné que le logiciel RATS possède la particularité de s'enrichir régulièrement de nouvelles procédures qui permettent la mise en œuvre des techniques statistiques les plus récentes dans le domaine de la modélisation des séries chronologiques. En particulier, l'analyse multivariée de séries chronologiques par la théorie de la cointégration (voir les articles de Granger (1986), Engle et Granger (1987), Johansen et

Juselius (1990, 1992, 1994)) a donné lieu à la mise au point de deux procédures extrêmement développées permettant l'application de la cointégration dans RATS. Ces deux procédures sont CATS (*Cointegration Analysis of Time Series*), développée par Henrik Hansen, et MALCOM, développée par Rocco Mosconi. Des renseignements sur ces procédures peuvent être obtenus, respectivement, aux adresses URL suivantes :

CATS : http://www.econ.ku.dk/hansen/
MALCOM : http://www.greta.it/malcom
http://www.ecopro.polimi.it/home/Rocco.Mosconi/

Les références suivantes concernent l'utilisation de CATS dans RATS :

Hansen, H. and Juselius, K. (1995), *CATS in RATS: Cointegration Analysis of Time Series*, Estima, Evanston, IL.

Tufte, D. (1998), "CATS in RATS: Cointegration analysis of time series: Version 1.01", *Journal of Applied Econometrics*, 13, 321-330.

Nous donnons enfin ci-dessous une liste non exhaustive de sites Internet mettant en ligne des procédures RATS.

→http://www.estima.com.
De nombreuses procédures et des exemples d'applications, sont postés sur le site de la société Estima, qui distribue le logiciel RATS. De plus, ce site propose une inscription gratuite à une mail-liste sur laquelle de nombreux utilisateurs de RATS viennent échanger diverses réponses et questions concernant l'utilisation du logiciel. Les archives de cette liste depuis 1996 sont disponibles à l'adresse URL suivante :
http://fmwww.bc.edu/ec-p/software/rats/ratsl.html.

→http://lo.ferrara.free.fr
Mon site personnel à partir duquel ce manuel peut être téléchargé, contenant également un document de cours pour l'économétrie des séries chronologiques à l'aide de processus linéaires et non linéaires. De plus, plusieurs procédures relatives aux processus longue mémoire sont disponibles.

→http://www.hec.ca/pages/simon.van-norden/
Page personnelle de Simon Van Norden (Ecole des Hautes Etudes Commerciales de Montréal)

→http://gsbwww.uchicago.edu/fac/ruey.tsay/teaching/fts/
Page relative au livre de R. Tsay intitulé « Analysis of Financial Time Series », contenant plusieurs procédures Rats.

→http://netec.mcc.ac.uk/CodEc/CodEc.html
CodEc (Programs for Economics and Econometrics) est un site qui rassemble des codes sources pour différents logiciels d'économétrie, en particulier RATS. CodEc est une partie du projet NetEC, initié par Thomas Krichel en 1993, qui comprend également comme services BibEc (une bibliographie de documents de travail en économie), WoPEc, (une base de données contenant des articles téléchargeables), et WebEc (liens hypertexte vers des sites web relatifs à l'économie). Pour d'autres renseignements sur le projet CodEc, on se réfère à l'article suivant :

Eddelbuttel, D. (1997), "A code archive for economics and econometrics", *Computational Economics*, 10, 4.

→ http://ideas.repec.org/
Ce site contient l'archive *Statistical Software Component* et contient de nombreuses procédures pour différents logiciels de statistique et d'économétrie. Il est maintenu à jour par Christopher Baum (Boston College, Department of Economics)

Enfin, les deux sites suivants proposent de nombreuses références utiles pour les statisticiens et les économètres, et en particulier quelques codes de méthodes statistiques, à traduire et à implémenter dans RATS, si besoin est.

→http://econometriclinks.com
De nombreux liens Internet sur tout ce qui touche à la statistique et l'économétrie (logiciels, codes sources, conférences, livres, jeu de données ...). Ce site est très complet et est maintenu à jour par Marius Ooms (Erasmus University Rotterdam), dont le site web se trouve à l'adresse suivante: http://www.eur.nl/few/ei/links/ooms/

→http://lib.stat.cmu.edu/
StatLib est un système géré par le Département de Statistique de l'Université Carnegie Mellon, qui permet la distribution en ligne de logiciels de statistiques (freewares et sharewares), de jeu de données et de nombreuses autres informations.

1. Premiers pas avec RATS

Dans cette partie, nous allons voir comment s'organise une session RATS de manière interactive, comment importer des données dans le logiciel et enfin voir comment faire apparaître ces données à l'écran.

Démarrage d'une session

Une session RATS interactive s'organise sur deux fenêtres différentes : une fenêtre qui sera utilisée en *input* et une autre qui sera utilisée en *output*. Sur la fenêtre d'*input*, l'utilisateur écrit les commandes du programme qu'il désire voir effectuer et les résultats apparaissent alors dans la fenêtre d'*output*. Il convient donc de préciser au préalable l'affection des fenêtres. Ainsi, lorsqu'on lance le logiciel RATS, une fenêtre nommée NONAME00.TXT apparaît. On donne alors un nom à ce fichier, par exemple ESSAI.PRG. Pour cela, il faut aller dans le menu 'File' et choisir l'option 'Save'. On note que l'extension .PRG est l'extension par défaut d'un fichier programme de RATS. On ouvre maintenant une seconde fenêtre, qui sera utilisée en *output*, à l'aide de l'option 'New' dans le menu 'File'. Pour affecter le rôle d'*output* à cette fenêtre, on choisit l'option 'Use for Output' dans le menu 'Window'. Enfin, il est agréable de travailler avec un écran partagé en deux, ce qui s'obtient par l'option 'Tile Horizontal' du menu 'Window'. L'utilisateur est maintenant prêt à taper dans la fenêtre ESSAI.PRG{I} les commandes qu'il désire voir s'effectuer dans la fenêtre NONAME00.TXT{O}. S'il le désire, l'utilisateur peut également sauvegarder la fenêtre d'*output* en lui donnant un nom.

Notons que RATS ne différencie pas les minuscules et les majuscules, et qu'une ligne de commande qui commence par le symbole suivant : *, n'est pas lue par le logiciel. De plus, le passage à la ligne d'un même ligne de commandes s'effectue à l'aide du signe $.

Allocation de la mémoire

Cette partie est extrêmement importante, car toute session RATS commence par une allocation de mémoire. Ceci se fait à l'aide des instructions `calendar` et `allocate`.

<u>Remarque :</u>

> Seules les trois premières lettres du nom des instructions sont nécessaires. Par exemple, l'instruction `calendar` s'effectue si on tape la commande `cal`. Cependant, il est recommandé, surtout pour les débutants avec RATS, de taper l'instruction en entier.

L'instruction `calendar` permet de fixer successivement l'année de début des observations, la période de début des observations et la périodicité des observations. L'instruction `allocate` permet de fixer la date de fin des observations. Par exemple, si on manipule pendant la session des séries trimestrielles commençant au premier trimestre 1990 et se terminant au troisième trimestre 1991, on tapera les instructions suivantes :

```
calendar 1990 1 4
allocate 1991:03
```

Le logiciel permet de travailler avec de nombreuses périodicités différentes, que l'on spécifie à l'aide de l'instruction `calendar`, et on se réfère au manuel RATS pour le détail de ces différentes périodicités. Notons que si l'utilisateur désire repérer les différentes observations uniquement à l'aide des nombres entiers, allant de 1 à la taille de l'échantillon, il n'est pas nécessaire d'utiliser l'instruction `calendar` seule l'instruction suivante est à passer :

```
allocate taille de l'échantillon
```

Par exemple, si on désire effectuer des simulations des séries de taille T=10000, il suffit de taper la commande suivante :

```
allocate 10000
```

Ecriture / Lecture des données

La lecture ou l'écriture des données dans le logiciel se fait à l'aide de l'instruction `data`. Dans la pratique, on travaille souvent avec des données stockées dans des fichiers extérieurs à RATS, de format Excel, Lotus, ... Notons qu'il existe également un format de fichier de type RATS (fichiers à extension .rat). Ces fichiers sont gérés par l'interface RATSData, qui permet de visualiser les données. L'import des données dans le logiciel se fait à l'aide de l'utilisation simultanée de l'instruction `open data`, qui permet d'indiquer au logiciel le fichier dans lequel aller chercher les données, et de l'instruction `data`, qui permet de lire les données. Par exemple, pour importer les données relatives à l'économie du Canada, contenues dans le fichier CANDATA.RAT, on effectue les commandes suivantes :

```
calendar 1960 1 12
all 1999 :12
open data 'c:\winrats\candata.rat'
data(format=rats) / CANM1S CANCD90D CANCPINF CANUSXSR
```

Ainsi, on a importé les séries mensuelles relatives au Canada contenues dans le fichier CANDATA.RAT, en les nommant après le signe / signifiant que l'on importe les données depuis la date de début (Janvier 1960) jusqu'à la date de fin (Décembre 1999) des séries. On remarque que les options d'une instruction sont écrites entre parenthèses, immédiatement après l'instruction, et sont séparées par une virgule. L'instruction `data` possède les options importantes suivantes :

`organization`	organise les données par variables (par défaut) ou par observations,
`format`	spécifie le format des données (Excel, Lotus, …)

Il arrive cependant quelques fois que l'on désire entrer soi-même les données "à la main" dans le logiciel. L'écriture des données se fait alors à l'aide de l'option `unit=input` de l'instruction `data`. Ainsi, si on désire entrer les valeurs de nouvelles séries pour les 6 premiers mois de l'année 1960, on tape au choix les commandes suivantes :

```
data(unit=input) 1960:01 1960:06 serie1 serie2
1 3 5 6 8 9
2 4 6 7 9 10
data(unit=input,org=obs) 1960:01 1960:06 serie1 serie2
1 2
```

```
3 4
5 6
6 7
8 9
9 10
```

Affichage des données

Certaines instructions permettent d'afficher les séries en cours d'utilisation. Par exemple, l'instruction `show series` liste le nom de toutes les séries présentes dans la session, et l'instruction `table` renvoie également des statistiques sur chacune des séries, à savoir le nombre d'observations, la moyenne, l'écart-type, le minimum et le maximum.
Pour afficher les valeurs des séries, on utilise la commande `print`, de la manière suivante :

print *date début date fin serie1 serie2 ... serien*

Par exemple, les commandes suivantes :

```
print / CANUSXSR CANM1S
print 89:01 89:12 CANUSXSR CANM1S
```

permettent d'afficher les valeurs des deux séries CANUSXSR et CANRGNP entre la date de début et la date de fin indiquées. La première commande affiche les valeurs de la séries sur l'ensemble de la période définie par les instructions `calendar` et `allocate`, et la seconde commande affiche les valeurs des séries uniquement pour l'année 1989. Les résultats obtenus par la seconde commande sont les suivants :

```
ENTRY       CANUSXSR         CANM1S
 1989:01       0.8452           41332
 1989:02       0.8342           40862
 1989:03       0.8381           41204
 1989:04       0.8431           40879
 1989:05       0.8294           41510
 1989:06       0.8345           41261
 1989:07       0.8471           41572
 1989:08       0.8505           42065
 1989:09       0.8487           41996
 1989:10       0.8517           42394
 1989:11       0.8597           41448
 1989:12       0.8637           42382
```

Remarque:

Dans toutes les instructions RATS, il est possible de spécifier la date de début et la date de fin d'exécution de l'instruction. Cependant, cette spécification alourdit un peu l'écriture du programme. Afin de pallier à ce problème, on utilise le slash, qui remplace automatiquement par défaut la spécification des dates de début et de fin par les dates spécifiées en début de programme par les instructions `calendar` et `allocate`. Par exemple, pour imprimer les séries CANUSXSR et CANM1S entre le mois de janvier 1960 et le mois de mars 1990, soit l'ensemble de la période, on aurait pu taper la commande suivante :

```
print / CANUSXSR CANM1S
```

La commande `smpl` permet de spécifier une période par défaut. Par exemple, si on veut travailler uniquement sur la période couvrant les années 1980, on effectuera la commande suivante :

```
smpl 80:01 89:12
print / CANUSXSR
smpl
```

La dernière commande `smpl` sans rien à la suite, permet de retourner à la période par défaut initialement déclarée en ouverture de session.

Les graphiques

RATS permet d'obtenir des graphiques d'excellente qualité, et les possibilités graphiques sont étendues. On se réfère au Chapitre 2 du manuel RATS pour une présentation détaillée, en particulier pour les nombreuses différentes options des instructions graphiques.
RATS possède deux instructions pour tracer des graphes : `graph` et `scatter`.

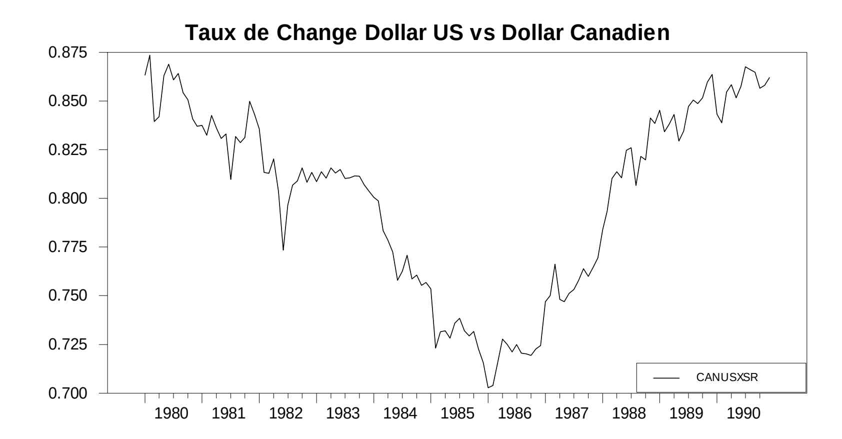
<u>L'instruction graph.</u>

Cette commande permet de représenter l'évolution d'une série temporelle, de la manière suivante :

```
graph(options) n
# serie1
# serie2
.....
# serien
```

Le paramètre n est le nombre de séries que l'on désire représenter sur le même graphique. On peut également spécifier les dates de début et de fin de la période souhaitée, le titre ou une légende, de la manière suivante :

```
graph(header="Taux de Change US Dollar / Dollar
Canadien",key=loright) 1
# CANUSXSR 80:01 89:12
```



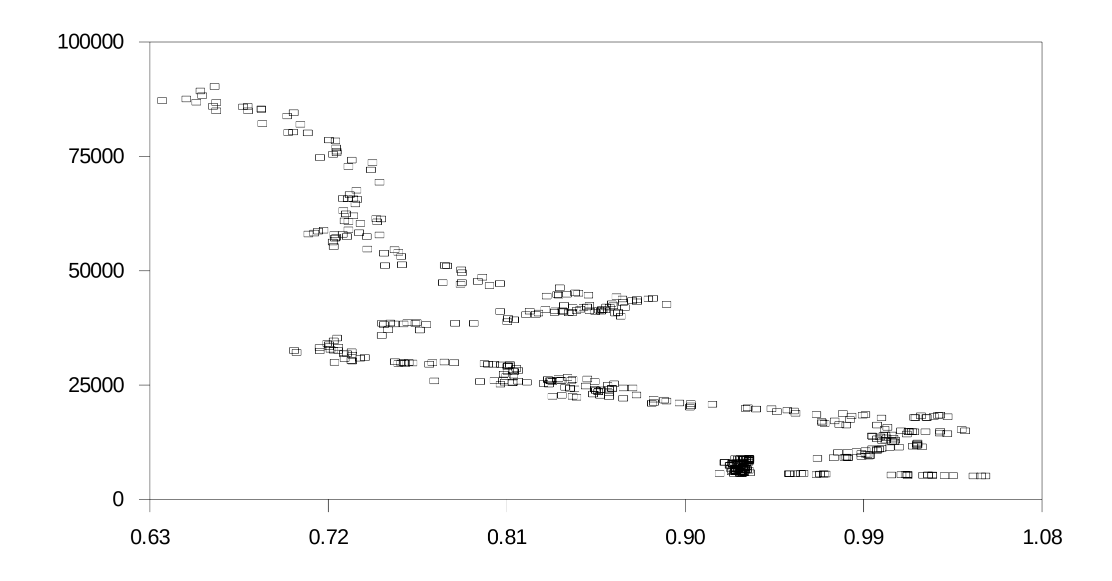
Les options les plus utilisées sont les suivantes :

`header=`	une chaîne de caractères placée entre guillemets
`key=`	permet d'insérer une légende dans le graphe, en la situant en bas à droite (low right →lor), en bas à gauche (low left →lol), en haut à droite (up right→upr) ou en haut à gauche (up left →upl).
`[dates]/nodates`	RATS met une légende en abscisse, sauf si l'option nodates est spécifiée
`patterns/[nopatterns]`	permet de distinguer les différentes courbes présentes sur le graphe à l'aide de traits différents, si l'option patterns est omise, RATS différencie les courbes par des couleurs.
`style=`	le style par défaut est une ligne, les autres styles utilisés sont `bar`, `vertical`, `step` et `symbols`.

L'instruction scatter

Cette instruction permet de représenter des points de coordonnées (X_i, Y_i) pour i=1,...,T, où X et Y sont deux séries. On utilise alors cette instruction de la manière suivante :

```
scatter 1
# canusxsr cantbill
```

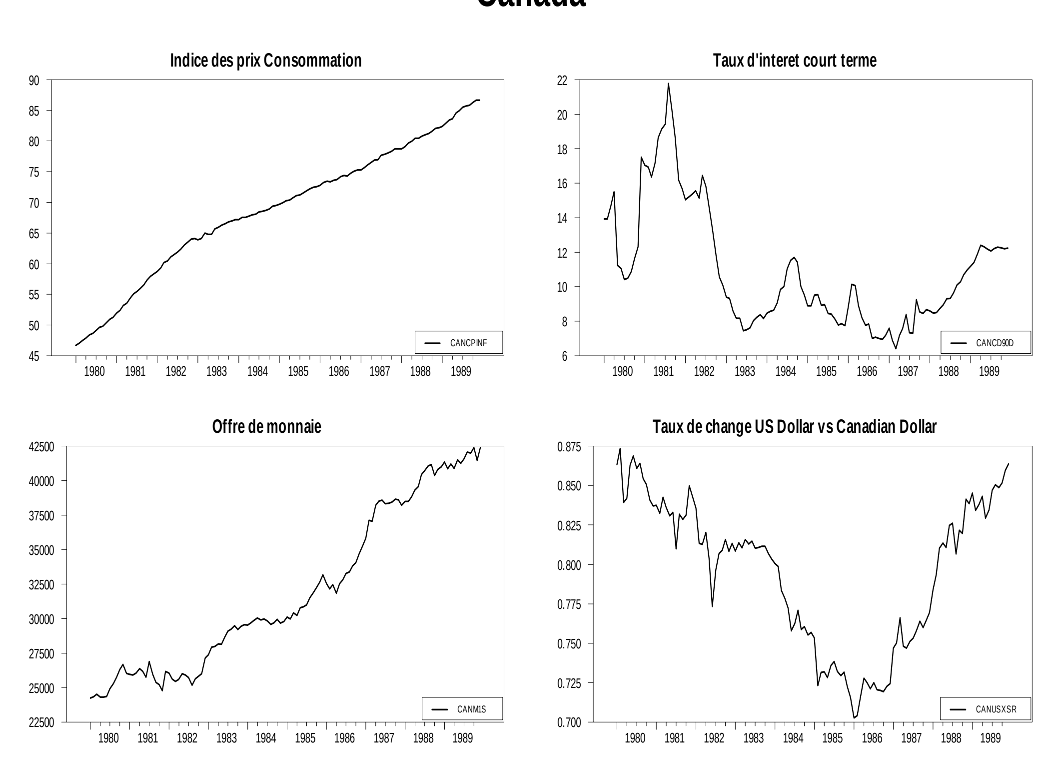


Les graphes multiples

RATS permet de tracer sur une même feuille graphique plusieurs graphes différents à l'aide de la commande `spgraph`, de la manière suivante :

```
spgraph(header="Canada",vfields=2,hfields=2)
graph(header="Indice des prix Consommation",key=lor) 1
```

```
# CANCPINF 80:01 89:12
graph(header="Offre de monnaie",key=lor) 1
# CANM1S 80:01 89:12
graph(header="Taux d'interet court terme ",key=lor) 1
# CANCD90D 80:01 89:12
graph(header="Taux de change US Dollar vs Canadian Dollar",key=lor) 1
# CANUSXSR 80:01 89:12
spgraph(done)
```



L'option `vfields` permet de choisir le nombre de champs verticaux et l'option `hfields` permet de choisir le nombre de champs horizontaux sur le graphique. On note que le graphe multiple doit se terminer par la commande `spgraph(done)`.

Les différents graphiques peuvent être sauvés sous différents format de type .RGF, .EPS, .WMF, .PLT ou .PIC, pour être ensuite intégrés dans un document de travail. En particulier, l'enregistrement d'un graphe au format .EPS, permet de l'intégrer facilement dans un document LaTex.

2. Opérations sur les séries

Dans cette partie, nous allons voir comment gérer des séries chronologiques et des scalaires, réels et entiers. Puis, nous présentons quelques fonctions statistiques intégrées au logiciel.

Création / Transformation d'une série

La création d'une nouvelle série et la transformation d'une série existante sont effectuées à l'aide de la commande `set`. La syntaxe est la suivante :

```
set y = formule
```

où `formule` peut prendre diverses formes, que nous allons détailler..

Par exemple, si on veut renommer une série préexistante, sur une période donnée, on effectue la commande suivante :

```
set y 80:01 89:12 = CANUSXSR
```

Si on désire élever au carré cette série :

```
set ycarre = y**2
```

L'opérateur ** permet d'élever à la puissance, les autres opérateurs mathématiques sont :

+	addition
-	soustraction
*	multiplication
/	division

Les autres fonctions mathématiques sont :

LOG	logarithme népérien
SQRT	racine carré
EXP	exponentielle
ABS	valeur absolue
COS, SIN, TAN	fonctions trigonométriques

Un autre opérateur sur les séries qui est très souvent utilisé dans l'étude des séries chronologiques est l'opérateur retard (backward), qui permet d'obtenir une série retardée dans le temps, défini tel que $BX_t=X_{t-1}$ et pour tout entier b, $B^bX_t=X_{t-b}$. On obtient une série retardée, avec un retard égal, par exemple, à b=1, de la manière suivante :

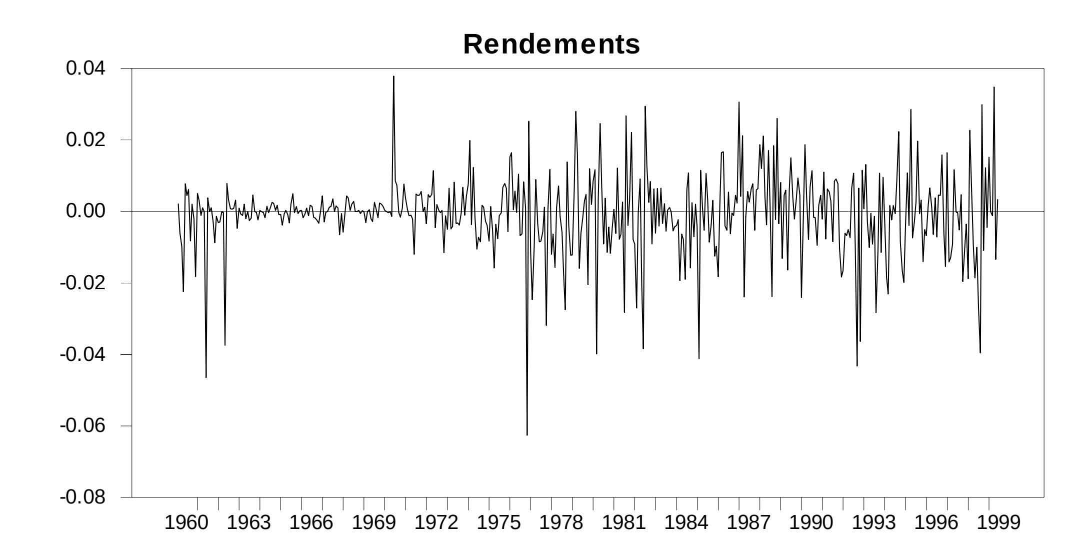
```
set y1 = CANUSXSR{1}
print / y1 CANUSXSR
```

De la même manière, pour obtenir une série avancée on effectue la même opération, en plaçant un signe moins devant le retard.

Dans de nombreuses séries chronologiques d'actifs financiers, on travaille sur les rendements (returns) de la série d'étude $(X_t)_t$, définis par $r_t=\log(X_t)- \log(X_{t-1})$. Par exemple,

la série des rendements du taux de change Dollar US / Dollar Canadien est obtenue par la commande :

```
set rdt = log(CANUSXSR)-log(CANUSXSR{1})
gra(header="Rendements")
# rdt
```



Cet opérateur retard peut être utilisé pour obtenir un lissage exponentiel simple d'une série. On rappelle que si on désire effectuer un lissage exponentiel simple, de paramètre α, d'une série d'étude $(X_t)_t$, la série lissée $(X^{le}_t)_t$, s'obtient alors par les égalités suivantes :

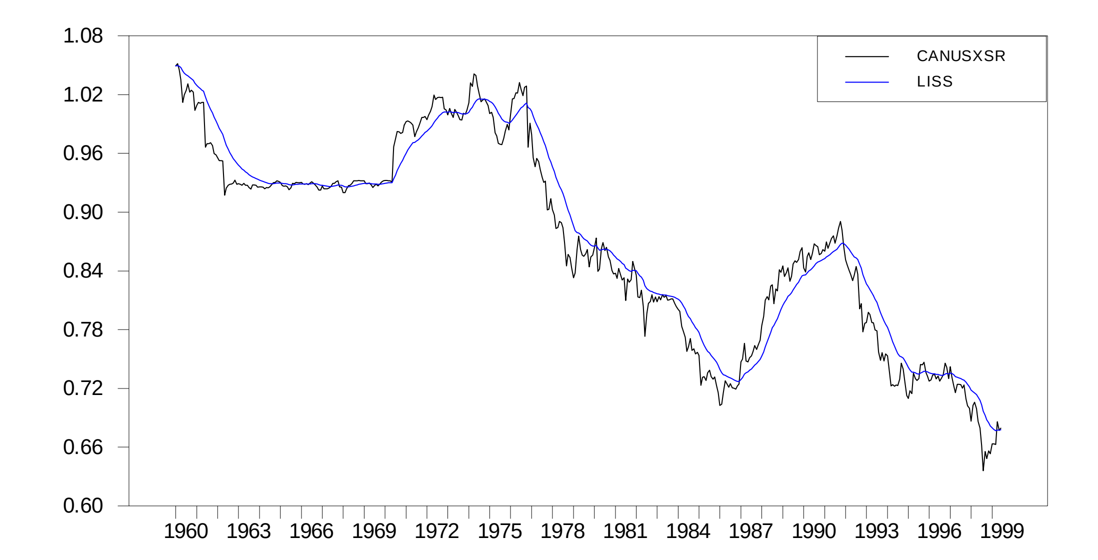
$$X^{le}_1 = X_1$$

et,

pour tout t>1, $$X^{le}_t = (1-\alpha)\, X^{le}_{t-1} + \alpha\, X_t$$

Par exemple, pour obtenir un lissage exponentiel simple de la série CANUSXSR, de paramètre α=0.1, on effectue les commandes suivantes :

```
set liss 60:01 60:01 = CANUSXSR(60:01)
set liss 60:02 99:12 = 0.9*liss{1}+0.1*CANUSXSR
graph(key=upr) 2
# CANUSXSR
# liss
```



Notons que pour les calculs récurrents, on peut également utiliser l'option `first` de la commande `set`, de la manière suivante :

```
set(first= CANUSXSR(60:01)) liss1 = 0.9*liss1{1}+0.1*CANUSXSR
```

Il est parfois utile de créer une série qui représente l'évolution du temps. Cette série, que l'on appelle, par exemple, date s'obtient par la commande suivante :

```
set date = t
```

Cette série vaut 1 pour la première date définie par l'instruction `calendar`, 2 pour la seconde date, *etc*...

Création d'une série aléatoire

Il est souvent intéressant de simuler des trajectoires de séries chronologiques, engendrées par un processus que l'on a déterminé au préalable. Pour cela, il est nécessaire d'engendrer des réalisations indépendantes issues d'une loi de distribution. RATS propose comme loi de distribution, la loi Normale et loi Uniforme. Les commandes sont les suivantes :

```
set seriesim1 = %RAN(x)
```

où x est l'écart-type de la loi Normale centrée que l'on désire obtenir, et

```
set seriesim2 = %UNIFORM(L,H)
```

où L et H sont les réels correspondant aux bornes de l'intervalle.
Si on désire reproduire exactement la même suite de nombre aléatoire, il faut alors utiliser l'instruction `seed`, de la manière suivante :

```
seed valeur
```

où `valeur` est un entier quelconque que l'on choisit, et qui détermine de façon unique la suite de nombre aléatoire. Par exemple, les commandes suivantes engendrent une suite de 10 nombres issus d'une loi Normale de variance égale à 4, référencée par l'entier 123 :

```
smpl 1 10
seed 123
set v1 = %ran(2)
```

On peut alors réutiliser cette même génération de nombre aléatoire, plus tard au cours de la même session ou même au cours d'une autre session RATS.

```
seed 123
set v2 = %ran(2)
print / v1 v2
```

Les séries v1 et v2 sont alors identiques. Notons que le choix de l'entier `valeur` n'a aucune influence sur la génération aléatoire, il sert simplement de référence.

Gestion des scalaires

Dans cette partie, on s'intéresse à la gestion par RATS des scalaires, réels et entiers. Les nombres complexes seront traités dans un chapitre à part.

Affectation

L'instruction d'affection d'une valeur à un scalaire est l'instruction, qui s'utilise de la manière suivante :

`compute(`*`options`*`)` *`scalaire résultat = formule de calcul ou valeur`*

Les options de cette instruction sont `real` (nombre réel) et `integer` (nombre entier). Cependant, ces options sont peu utilisées car, si la formule contient un nombre réel, alors le résultat sera un nombre réel. De même, lors de la définition d'un scalaire, si la valeur est un nombre décimal, alors le résultat sera un réel et si la valeur est un nombre entier, alors le résultat sera un entier. Par exemple, si on tape

```
com alpha = 0.8
```

alors, `alpha` sera un réel, et si on tape

```
com max = 12
```

alors `max` sera un nombre entier. Par contre, si on tape

```
com(real) max = 12
```

alors `max` sera un nombre réel.

Il est à noter que les dates sont gérées comme étant des entiers. On peut ainsi affecter des noms à des dates de la manière suivante :

```
com debut = 80:01
com fin = 89:12
```

Notons également que l'addition, la soustraction, la multiplication et la division des scalaires entre eux se fait à l'aide des opérateurs respectifs : +, -, *, et /.

Affichage

Pour afficher un scalaire, on utilise la commande `display` de la manière suivante :

`display` *`nom du scalaire`*

Par exemple, pour afficher la valeur de la série à une date donnée, on tape la commande :

```
display CANUSXSR(90:01)
```

L'instruction `display` permet également, soit d'afficher le résultat d'un calcul, par exemple:

```
display log(10)
display 2*cos(%PI/6)+1
```

où `%PI` est la valeur de π=3.14159...., soit d'afficher un texte, par exemple :

```
dis 'Le résultat est :'
```

On note alors que le texte doit être placé entre quotes.

Variables muettes

Une variable muette ou indicatrice (dummy) est une série qui prend pour valeur 0 sur certaines périodes et qui prend pour valeur 1 sur d'autres périodes. Comme nous le verrons plus tard, ce type de variable est utile dans de nombreux cas. On définit une variable muette, à l'aide de l'instruction `set`, par la commande suivante :

```
set dum 60:01 79:12 = 1.0
set dum 80:01 99:12 = 0.0
```

L'instruction seasonal permet de créer une variable muette saisonnière qui prend pour valeur 1 à certaines périodicités saisonnières et qui prend pour valeur 0 ailleurs. Par exemple, si on désire créer une série qui vaut 1 tous les mois de février et 0 partout ailleurs, à partir du mois de février 1960, on tape la commande suivante :

```
seasonal dums / 12 60:02
print / dums
```

Différentiation et filtrage linéaire d'une série

Nous verrons par la suite que de nombreuses méthodes de modélisation de séries chronologiques ne s'appliquent que sur des séries qui sont stationnaires. Malheureusement,

en pratique, de nombreuses séries ne sont pas stationnaires, car elles possèdent, par exemple, une tendance ou une saisonnalité. Il convient donc de transformer au préalable ces séries afin de les rendre asymptotiquement stationnaires. Une manière classique de procéder est de différencier ces séries. Pour cela on utilise l'instruction `difference`, de la manière suivante :

```
difference(options) série  début  fin  nouvelle série
```

Par exemple, pour obtenir la différence première de la série CANUSXSR, que l'on appelle DCANUSXSR, on effectue la commande suivante :

```
difference CANUSXSR / DCANUSXSR
```

On peut obtenir les différences d'ordre supérieur à l'aide de l'option `differences`. De même, cette instruction permet de différencier saisonnièrement une série à l'aide de l'option `sdiffs`, de la manière suivante :

```
difference(sdiffs=1) CANUSXSR / D12CANUSXSR
```

Par défaut, le logiciel utilise la périodicité déterminée en début de session par l'instruction `calendar`, sinon il faut spécifier la périodicité à l'aide de l'option `span`.

D'une manière générale, pour filtrer linéairement une série on utilise l'instruction `filter`, de la manière suivante :

```
filter(options) série  début  fin  nouvelle série
# liste des retards ou avances du filtre
# coefficients du filtre
```

Par exemple, si on désire calculer la série $Y_t = X_t - 0.5\ X_{t-1} + 0.2\ X_{t-2}$, où (X_t) représente la série CANUSXSR, on effectue la commande suivante :

```
filter CANUSXSR / FCANUSXSR
# 0 1 2
# 1.0 -0.5 0.2
```

Pour différencier les avances et les retards, on fait précéder les avances d'un signe moins. Par exemple, si on désire calculer la moyenne mobile centrée pondérée d'ordre 4 suivante :
$Y_t = (0.5\ X_{t-2} + X_{t-1} + X_t + X_{t+1} + 0.5\ X_{t+2},)/4$
on effectue alors la commande suivante :

```
filter CANUSXSR / MMCANUSXSR
# -2 -1 0 1 2
# 0.125 0.25 0.25 0.25 0.125
```

Une autre manière de filtrer une série est d'utiliser l'opérateur retard. Ainsi, la série FCANUSXSR peut également être obtenue de la manière suivante :

```
set FCANUSXSR = CANUSXSR-0.5*CANUSXSR{1}+0.2*CANUSXSR{2}
```

Quelques fonctions statistiques

L'instruction `statistics`

Cette instruction permet d'obtenir certaines statistiques de base pour une seule série, à savoir les moments d'ordres inférieurs ou égaux à 4 : moyenne, variance, écart-type, skewness et kurtosis. De plus, l'instruction `statistics` renvoie également les résultats du test de nullité sur la moyenne, le skewness et le kurtosis, ainsi que le test de jarque-Bera d'adéquation à la loi Normale, à travers les valeurs de la P-value. On rappelle que l'hypothèse H_0 de nullité est acceptée au risque α=0.95, si la P-value est supérieure à 1-α. Cette instruction fournit des résultats plus détaillés que l'instruction `table`, mais sur une seule série, alors que l'instruction `table` permet d'obtenir des statistiques sur plusieurs séries simultanément. L'instruction `statistics` s'utilise de la manière suivante :

statistics(*options*) *série début fin*

L'option `fractiles` permet d'obtenir également la médiane, le minimum, le maximum et les fractiles suivants : 1%, 5%, 10%, 25%, 75%, 90%, 95%, 99%. L'option `noprint` permet de ne pas afficher les résultats et l'option `nomoments` permet de ne pas afficher les valeurs des différents moments de la série. Par exemple, la commande :

```
sta CANUSXSR
```

renvoie en sortie :

```
Statistics on Series CANUSXSR
Monthly Data From 1960:01 To 1999:06
Observations   474
Sample Mean    0.86804852321          Variance                0.010659
Standard Error 0.10324036368          SE of Sample Mean       0.004742
t-Statistic      183.05586            Signif Level (Mean=0) 0.00000000
Skewness          -0.18482            Signif Level (Sk=0)   0.10151913
Kurtosis          -1.08422            Signif Level (Ku=0)   0.00000173
Jarque-Bera       25.91535            Signif Level (JB=0)   0.00000236
```

A la suite de cette instruction, de nombreux résultats sont conservés en mémoire par le logiciel. Ces résultats ont des noms précis qui commencent toujours par le symbole : %. Par exemple, suite à l'instruction `statistics`, on a accès aux scalaires suivants :
`%mean, %variance, %nobs, %cdstat, %skewness, %kurtosis, %median, %maximum, %minimum,` *etc* ...
Par exemple, la somme des termes de la série CANUSXSR peut être obtenue par la commande suivante :

```
dis %mean*%nobs
```

De nombreuses autres instructions RATS gardent ainsi en mémoire des résultats relatifs aux commandes que l'on vient d'effectuer, permettant ainsi de récupérer ces résultats pour les intégrer dans d'autres calculs.

L'instruction `extremum`

Cette instruction permet d'obtenir les valeurs extrêmes d'un série de la manière suivante :

```
extremum(options) série  début  fin
```

Cette série permet d'accéder aux scalaires suivants :

`%MAXENT`	date à laquelle le maximum est atteint (entier)
`%MAXIMUM`	valeur du maximum (réel)
`%MINENT`	date à laquelle le minimum est atteint (entier)
`%MINIMUM`	valeur du minimum (réel)

L'instruction `order`

Cette instruction permet de trier une série par ordre croissant ou décroissant (option decrease), de la manière suivante :

```
order(options) série  début  fin
```

Mais cette instruction permet également de trier une liste de séries en fonction des valeurs d'une série. Par exemple, si on désire trier les séries `CANM1S CANTBILL CANCPINF` en fonction des valeurs de la série `CANUSXSR` sur l'ensemble de la période, on effectue la commande suivante :

```
order CANUSXSR / CANM1S CANTBILL CANCPINF
```

L'instruction `accumulate`

Cette instruction permet de calculer les sommes partielles d'une série, de la manière suivante :

```
accumulate série  début  fin  cumserie
```

où `cumserie` est la série cumulée. Ainsi, si on effectue la commande

```
accumulate CANUSXSR / CUMCANUSXSR
```

alors la valeur `CUMCANUSXSR(79:12)` sera la somme de la série `CANUSXSR` de janvier 1960 à décembre 1979.

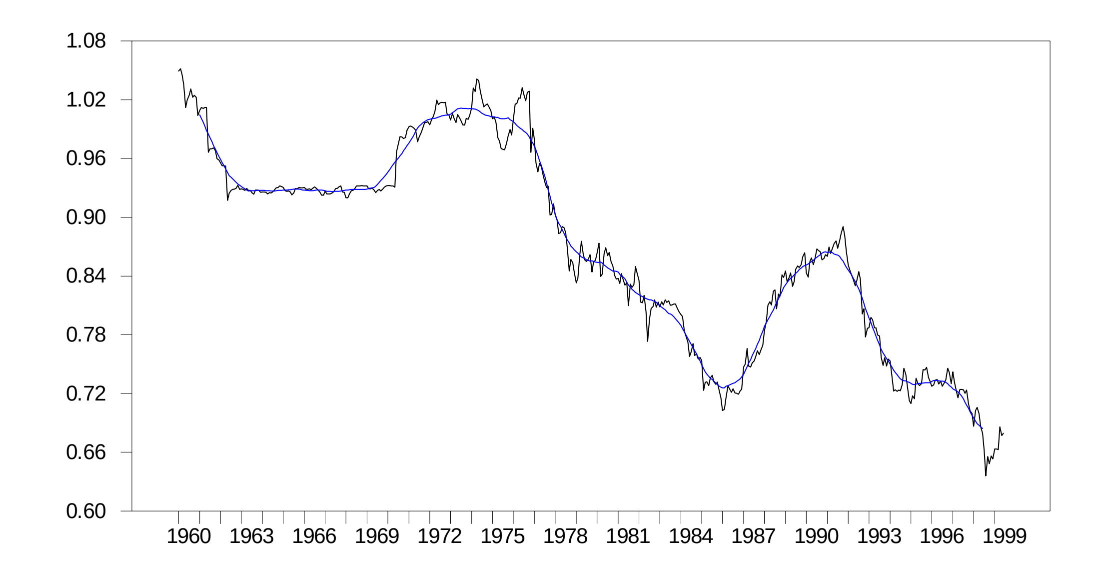
L'instruction `mvstats`

Cette instruction permet de calculer la série des moyennes mobiles et/ou des variances mobiles sur une série d'intérêt, de la manière suivante :

```
mvstats série  début  fin
```

Cette instruction ne renvoie pas de séries de sortie, mais elle permet de stocker les séries résultantes à l'aide des options `means` et `variances`. De plus, l'utilisateur peut choisir le nombre de points à inclure dans le calcul, à l'aide de l'option `span`, et il peut également choisir de centrer ou non la moyenne (ou la variance) mobile, à l'aide de l'option `centered`. Dans le cas où la moyenne mobile est centrée, le nombre `span` doit être impair. Par exemple, si on désire calculer la moyenne mobile centrée de la série mensuelle CANUSXSR, en prenant en compte le 12 mois avant et les 12 mois après, on tape alors les commandes suivantes :

```
mvstats(means=CANUSXSRMM,span=25,centered) CANUSXSR /
graph 2
# CANUSXSR
# CANUSXSRMM
```



Toutefois, notons que cette instruction ne permet pas de calculer des moyennes mobiles pondérées. Il faut alors utiliser l'instruction `filter`.

Il est également possible d'obtenir une série de fractiles mobiles, de la même manière que précédemment en utilisant l'instruction `mvfractile`.

<u>L'instruction `sample`</u>

Cette instruction permet de créer une sous-série, en échantillonnant à intervalles réguliers une série initiale. Cette instruction s'utilise de la manière suivante :

sample(interval=) *série début fin nouvelle série*

Par exemple, si on désire créer la sous-série des mois de février pour la série CANUSXSR, à partir du mois de février 1960, on tape alors les commandes suivantes :

```
sample(interval=12) CANUSXSR 60:02 90:03 CANUSXSRFEV
```

3. Opérations sur les tableaux

Le logiciel RATS permet de manipuler différents objets, autres que les séries chronologiques et les scalaires. En particulier, il permet la gestion de vecteurs et de matrices.

Gestion des tableaux

<u>Déclaration</u>

Pour manipuler les tableaux, il faut les déclarer au préalable, contrairement aux séries ou aux scalaires. La déclaration se fait par l'instruction `declare`, de la manière suivante :

`declare` *`type  liste des noms des tableaux avec leur dimension`*

Il existe 3 types différents de tableaux :

`REC`	matrices rectangulaires ou carrées
`SYM`	matrices symétriques (carrées)
`VEC`	vecteurs

Par exemple, la déclaration et le dimensionnement se font de la manière suivante :

```
declare rec MR1(3,2) MR2(4,4)
declare sym MS(2,2)
declare vec V(3)
```

Pour les matrices, la première dimension représente le nombre de lignes et la seconde représente le nombre de colonnes. L'opération de déclaration et l'opération de dimensionnement peuvent être effectuées successivement. Ceci peut s'avérer utile dans une procédure pour laquelle la dimension d'un vecteur est connue ultérieurement à sa déclaration. Par exemple, on aurait pu effectuer les commandes suivantes :

```
declare rec MR1 MR2
declare sym MS
declare vec V
dim MR1(3,2) MR2(4,4) MS(2,2) V(3)
```

<u>Affectation</u>

L'instruction d'affectation de valeurs à un tableau se fait de manière identique à l'affectation d'une valeur à un scalaire, à l'aide de l'instruction `compute`. Cependant, la procédure d'affectation obéit aux règles suivantes :
- une double barre pour délimiter un tableau,
- une simple barre pour délimiter une ligne d'un tableau
- une virgule pour délimiter les éléments d'une ligne
- les entrées se font par ligne

Par exemple, si on désire entrer les tableaux suivants :

V=(2,4,6), $MR1 = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$, $MS = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}$,

on tape alors les commandes suivantes :

```
com V=||1,2,3||
com MR1=||1,4|2,5|3,6||
com MS=||1|2,4||
```

Affichage

Pour afficher un tableau, on utilise l'instruction `write`, de la manière suivante :

```
write tab1  tab2  ... tabn
```

Par exemple, pour afficher les 3 tableaux précédents :

```
write V MS MR1
```

Génération de tableaux

On peut engendrer des tableaux soit à l'aide de l'instruction ewise, soit à l'aide de l'instruction `make`.

L'instruction ewise permet d'engendrer un tableau à l'aide d'indices et d'une formule de calcul. Ainsi, pour engendrer un vecteur, on tape, par exemple, la commande suivante :

```
ewise V(i) = i**2
```

On obtient alors le vecteur suivant :

```
write V
1.0000         4.0000         9.0000
```

Pour engendrer une matrice, on tape, par exemple, la commande suivante :

```
ewise MR2(i,j) = i+j
write MR2
2.0000         3.0000         4.0000         5.0000
3.0000         4.0000         5.0000         6.0000
4.0000         5.0000         6.0000         7.0000
5.0000         6.0000         7.0000         8.0000
```

L'instruction make permet d'engendrer un tableau à partir de séries existantes. Chaque série devient alors une colonne du tableau et le nombre de lignes du tableau est égal au nombre d'observations. L'instruction s'utilise selon la syntaxe suivante :

```
make(options) M /
# serie1 serie2 ... serien
```

où `M` est la matrice résultante. Par exemple, on peut créer la matrice `CANADA` à partir des séries `CANM1S, CANTBILL, CANCPINF` et `CANUSXSR`, de la manière suivante :

```
make canada /
# CANM1S CANTBILL CANCPINF CANUSXSR
```

Inversement, on peut recréer une série à partir d'un tableau. Dans le cas d'un vecteur, on tape la commande suivante :

```
set vserie = V(t)
print / vserie
ENTRY          VSERIE
 1960:01  1.0000000000000
 1960:02  4.0000000000000
 1960:03  9.0000000000000
```

Dans le cas d'une matrice, notée M, on peut créer une série, soit à partir de la $i^{ème}$ ligne, de la manière suivante :

```
set Mserie = M(i,t)
```

soit à partir de la $j^{ème}$ colonne, de la manière suivante :

```
set Mserie = M(t,j)
```

Par exemple, si on veut créer une série à partir de la deuxième ligne de la matrice MR2, on tape la commande :

```
set MR2serie = MR2(2,t)
print / MR2serie
ENTRY         MR2SERIE
 1960:01  3.0000000000000
 1960:02  4.0000000000000
 1960:03  5.0000000000000
 1960:04  6.0000000000000
```

On peut également engendrer des tableaux à l'aide des fonctions RATS suivantes :

- `%identity(n)` permet de créer une matrice identité de dimension n,
- `%mscalar(x)` permet de remplir une matrice avec le réel x sur la diagonale,
- `%const(x)` permet de remplir une matrice à éléments identiques.

Par exemple, on utilise ces 3 fonctions comme suit :

```
com A = %identity(3)
write A
1.0000         0.0000         0.0000
0.0000         1.0000         0.0000
0.0000         0.0000         1.0000
```

```
dec rec B(3,3)
com B = %mscalar(2.5)
write B
2.5000          0.0000          0.0000
0.0000          2.5000          0.0000
0.0000          0.0000          2.5000

dec rec C(3,2)
com C = %const(2.5)
write C
2.5000          2.5000
2.5000          2.5000
2.5000          2.5000
```

On remarque que pour la première fonction, il n'est pas nécessaire de déclarer au préalable la matrice.

On peut enfin engendrer des tableaux aléatoires à partir d'une loi de distribution. Par exemple, on remplit une matrice A, dimensionnée au préalable, avec des réalisations issues d'une loi Normale centrée d'écart-type x, de la manière suivante :

```
com A = %ran(x)
```

De même on peut remplir cette matrice A avec des réalisation issues d'une loi Uniforme sur l'intervalle [x1,x2], de la manière suivante :

```
com A = %uniform(x1,x2)
```

Eléments de calcul matriciel

L'instruction compute permet de faire des calculs en combinant les tableaux et les scalaires, selon la syntaxe suivante :

compute *tableau résultat = formule de calcul*

On peut ainsi additionner (opérateur "+") ou soustraire (opérateur "-") 2 tableaux de dimensions identiques, par exemple :

```
com AB = A+B
write AB
3.5000          0.0000          0.0000
0.0000          3.5000          0.0000
0.0000          0.0000          3.5000
```

Mais si les dimensions ne sont pas identiques, on obtient alors le message suivant :

```
write A+C
## MAT2. Matrices with Dimensions 3 x 3 and 3 x 2 Involved in +
Operation
```

On peut également multiplier deux tableaux de dimension identiques à l'aide de l'opérateur "*", de la manière suivante :

```
write A*B
2.5000          0.0000          0.0000
0.0000          2.5000          0.0000
0.0000          0.0000          2.5000
```

Il faut noter que lorsqu'on multiplie une matrice par un vecteur, de dimension compatible, le résultat n'est pas un vecteur pour RATS, mais une matrice.

La multiplication d'un tableau et d'un scalaire se fait de manière identique, à l'aide de l'opérateur "*", mais pour diviser un tableau par un scalaire il faut multiplier par l'inverse du scalaire. Par contre, les opérations d'addition et de soustraction entre un scalaire et un tableau ne peuvent pas s'effectuer directement. L'astuce consiste à engendrer un tableau de valeurs identiques à partir du scalaire. Par exemple, si on désire ajouter la valeur 2 au tableau A, on effectue les commandes suivantes :

```
dec rec CSTE(3,3)
com CSTE = %const(2.0)
write CSTE
write A+CSTE
3.0000          2.0000          2.0000
2.0000          3.0000          2.0000
2.0000          2.0000          3.0000
```

Pour transformer tous les termes d'un tableau par des fonctions mathématiques classiques, on utilise les fonctions suivantes :

`%log(A)`	renvoie la matrice des logarithmes des termes de A
`%exp(A)`	renvoie la matrice des exponentiels des termes de A
`%abs(A)`	renvoie la matrice des valeurs absolues des termes de A
`%sqrt(A)`	renvoie la matrice des racines carrées des termes de A

Quelques fonctions matricielles

RATS possède des fonctions intégrées qui renvoient des valeurs calculées sur les matrices et les vecteurs; nous en donnons quelques unes.

`inv(A)`	renvoie l'inverse de A (matrice carrée)
`tr(A)`	renvoie la transposée de A (matrice)
`%rows(A)`	renvoie le nombre de lignes de la matrice A (entier)
`%cols(A)`	renvoie le nombre de colonnes de la matrice A (entier)
`%xcol(A,j)`	renvoie la $j^{ème}$ colonne de la matrice A (matrice $n \times 1$)
`%xrow(A,i)`	renvoie la $i^{ème}$ ligne de la matrice A (matrice $n \times 1$)
`%xdiag(A)`	renvoie la diagonale de la matrice A (matrice $n \times 1$)
`%decomp(A)`	décomposition de Choleski d'une matrice symétrique définie positive. (matrice triangulaire inférieure S telle que SS'=M)
`%diag(A)`	renvoie une matrice diagonale crée à partir du vecteur M
`%mqform(A,B)`	renvoie B'AB, où A est $N\times N$ et B est $N\times M$ (matrice)
`%qform(A,B)`	renvoie B'AB, où A est $N\times N$ et B est $N\times 1$ (vecteur)
`%mqformdiag(A,B)`	renvoie seulement la diagonale de B'AB (vecteur)

`%det(A)`	renvoie le déterminant de la matrice carrée A (réel)
`%trace(A)`	renvoie la trace d'une matrice carrée (réel)
`%cov(A,B)`	renvoie la covariance entre A et B (réel) (en général A et B sont des vecteurs)
`%corr(A,B)`	renvoie la corrélation entre A et B (réel) (en général A et B sont des vecteurs)
`%scalar(A)`	renvoie le terme `A(1,1)` si A est une matrice ou `A(1)` si A est un vecteur (réel)
`%sum(A)`	renvoie la somme des éléments de A (réel)
`%maxvalue(A)`	renvoie le maximum des valeurs de A (réel)
`%minvalue(A)`	renvoie le minimum des valeurs de A (réel)

Il est à noter que ces trois dernières fonctions sont également valables dans le cas où A est une série.

Autres instructions matricielles

L'instruction eigen permet d'obtenir les valeurs propres et les vecteurs propres d'une matrice carrée, de la manière suivante :

```
eigen(options) tableau  valeurs propres  vecteurs propres
```

où valeurs propres est le nom que l'on donne au vecteur des valeurs propres du tableau et vecteurs propres est le nom que l'on donne à la matrice des vecteurs propres.

4) Outils de programmation

On se réfère dans cette partie au Chapitre 4 du manuel d'utilisation RATS (Doan (1992), intitulé "Programmer's Tools". Les outils de programmation permettent à l'utilisateur d'écrire ses propres procédures avec RATS.

Les opérateurs logiques

On donne une liste des opérateurs logiques.

`X.eq.Y`	ou	`X==Y`	X est égal à Y
`X.ne.Y`	ou	`X<>Y`	X est différent de Y
`X.gt.Y`	ou	`X>Y`	X est strictement supérieur à Y
`X.ge.Y`	ou	`X>=Y`	X est supérieur ou égal à Y
`X.lt.Y`	ou	`X<Y`	X est strictement inférieur à Y
`X.le.Y`	ou	`X<=Y`	X est inférieur ou égal à Y
`.not.X`			X est faux
`X.and.Y`			X et Y sont vraies
`X.or.Y`			X ou Y sont vraies

Par exemple, pour créer des variables muettes, on effectue les commandes suivantes :

`set dum1 = T==1980:01`	la série dum1 vaut 1 en janvier 1980 et 0 ailleurs
`set dum2 = T<=1979:12`	la série dum2 vaut 1 jusqu'en décembre 1979 et 0 après
`set dum3 = T<=79:12.and.T>=81:01`	la série dum3 vaut 0 pour l'année 1980 et 1 ailleurs

Les boucles de calcul

RATS offre cinq types différents de boucles.

<u>La boucle do</u>

C'est la boucle standard, qui permet d'exécuter un même groupe d'instruction selon un indice que l'on incrémente systématiquement. Sa structure est la suivante :

```
do indice=valeur initiale, valeur finale, incrément
{
liste des instructions à effectuer
}
end do
```

Par défaut, la valeur de l'incrément est de 1. Par exemple, pour calculer (de manière peu efficace) la somme de la série CANUSXSR pour l'année 1980, on effectue la commande suivante :

```
com somme=0.0
```

```
do i=80:01,80:12
{
com somme=somme+CANUSXSR(i)
}
end do i
dis somme
```

Notons qu'on aurait pu effectuer de manière plus efficace, les commandes suivantes :

```
sta CANUSXSR 80:01 80:12
dis %mean*%nobs
```

D'une manière générale, les boucles de calcul sont utiles, mais elles alourdissent l'écriture et posent des problèmes de mémoire, en particulier si on effectue des boucles imbriquées. Ainsi, lorsqu'on effectue une boucle, il est bon de se demander au préalable s'il n'existe pas une manière plus efficace de programmer.

La boucle `dofor`

Elle permet d'exécuter plusieurs fois un même groupe d'instructions selon :
- une liste de valeurs entières (par défaut) ou réelles
- une liste de variables

Si les valeurs de la liste sont réelles, il faut déclarer la variable au préalable, à l'aide de la commande `dec`. Sa structure est la suivante :

```
dofor indice=liste de valeurs (ou variables)
{
liste des instructions à effectuer
}
end dofor
```

Par exemple, pour obtenir uniquement les fractiles des séries CANM1S, CANTBILL, CANCPINF et CANUSXSR on effectue les commandes suivantes :

```
dofor i=CANM1S CANTBILL CANCPINF CANUSXSR
{
sta(nomoments,fractiles) i
}
end dofor
```

La boucle `while`

Elle permet d'effectuer une boucle tant qu'une certaine condition est vérifiée. Sa structure est la suivante :

```
while condition
{
liste des instructions à effectuer
}
end while
```

Par exemple, si on désire compter le nombre de valeurs supérieures ou égales à la valeur moyenne de la série CANUSXSR, avant de rencontrer une valeur inférieure à cette moyenne, on effectue les commandes suivantes :

```
sta CANUSXSR
com moyen = %mean; com j = 60:01; com cpt = 0
while CANUSXSR(j)>=moyen
{
com j=j+1
com cpt=cpt+1
}
end while
dis cpt
```

La boucle `until`

Elle permet d'effectuer une boucle jusqu'à ce qu'une certaine condition soit vérifiée. Sa structure est la suivante :

```
until condition
{
liste des instructions à effectuer
}
end until
```

La différence avec une boucle `while`, est que, dans le cas d'une boucle `until`, le test de la condition est effectué en bas de la boucle. Par conséquent, dans une boucle `until`, la liste des instructions est effectuée au moins une fois.

La boucle `loop`

Elle permet d'effectuer une boucle non conditionnelle. On sort de cette boucle à l'aide de l'instruction break. En général, on l'utilise de manière conjointe avec l'instruction menu, qui permet d'effectuer des choix de manière interactive. Par exemple, les commandes suivantes affichent une fenêtre qui permet à l'utilisateur d'obtenir des statistiques sur une des séries au choix :

```
loop
menu 'Statistiques sur une serie'
   choice 'Serie CANUSXSR'
     sta CANUSXSR
   choice 'Serie CANM1S'
     sta CANM1S
   choice 'Serie CANTBILL'
     sta CANTBILL
   choice 'Exit'
     break
end menu
end loop
```

L'instruction if-else

Cette instruction permet d'exécuter une commande si une certaine condition est vérifiée. Sa structure est la suivante :

```
if condition 1
{liste des instructions à effectuer si condition 1 est vraie}
else if condition n
{liste des instructions à effectuer si condition n est vraie et
aucune des conditions précédentes n'est vraie}
else
{liste des instructions à effectuer si aucune des conditions
précédentes n'est vraie}
end if
```

Par exemple, si on désire créer une série, de telle sorte que cette série prenne pour valeurs les valeurs de la série CANUSXSR si ces valeurs sont supérieures ou égales à 100, et 0 sinon :

```
set x1 = %NA
do i=60:01,90:03
      if CANUSXSR(i)>=100
      {
            com x1(i) = CANUSXSR(i)
      }
      else
      {
            com x1(i) = 0.0
      }
enddo i
print / CANUSXSR x1
```

Les procédures

Les procédures constituent un des outils les plus puissants de RATS. Elles permettent de définir de nouvelles instructions à partir d'une suite de commandes. Ces procédures sont stockées dans des fichiers texte à extension .SRC. Le logiciel en possèdent quelques unes, et de nombreuses autres autres peuvent être téléchargées à partir des sites web indiqués en introduction. L'import de la procédure dans le fichier .PRG, à partir duquel se fait la session RATS, est réalisée à l'aide de l'instruction source suivi du chemin dans lequel se trouve le fichier .SRC. L'appel de la procédure dans RATS se fait en faisant précéder le nom de la procédure par le signe : @.

Avant de se lancer dans l'écriture d'une procédure, il est fortement conseillé d'observer attentivement la manière dont sont structurées les procédures fournies par le logiciel. Basiquement, une procédure RATS commence par l'instruction `procedure` et se termine par l'instruction `end procedure`. Une procédure possède les traits caractéristiques suivants:

- les paramètres : ils sont placés sur la même ligne que le nom de la procédure et constituent l'information que l'on désire fournir en entrée. Leurs types sont spécifiés par l'instruction `type`.

- les variables locales : elles sont utilisées uniquement à l'intérieur de la procédure et elles sont spécifiées par l'instruction `local`

- les options : elles fonctionnent comme les autres options des instructions RATS et elles sont spécifiées par l'instruction `option`.

- les entrées supplémentaires : elles sont spécifiées par l'instruction `enter`.

Par exemple, nous allons créer une procédure qui renvoie uniquement les moments d'ordre 3 et 4 d'une série et qui, en option par défaut, permet d'afficher et de tracer la série. On appelle cette procédure `essai`. On ouvre donc un fichier texte qu'on enregistre sous le nom de ESSAI.SRC et on tape le texte suivant :

```
procedure essai xserie start end

* Déclaration des variables en entrée
type series       xserie
type integer      start end

* Spécification des options
option switch     graph       1
option switch     print       1

* Déclaration des variables locales
local real        skew kurto

* Calcul des moments d'ordre 3 et 4
sta(noprint) xserie start end
com skew = %skewness
com kurto = %kurtosis

* Affichage des résultats
dis 'Le Skewness est égal à :' skew
dis 'Le Kurtosis est égal à :' kurto

* Option d'affichage de la série
if print
{
print start end xserie
}

* Option de traçage de la série
if graph
{
graph
# xserie start end
}

end procedure
```

L'appel de la procédure dans RATS se fait alors de la manière suivante :

```
source(noecho) 'c:\winrats\essai.src'
@essai(noprint,graph) CANUSXSR 60:01 90:03
```

On voit à partir de cet exemple simple que l'écriture d'une procédure est relativement facile, et que l'on peut créer ses propres procédures au gré de ses besoins. On note en remarque,

qu'il vaut mieux donner le même nom au fichier .SRC et à la procédure qu'il contient, bien que cela ne soit pas obligatoire.